Министерство образования и науки Украины
Национальный технический университет
"Харьковский политехнический институт"

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным занятиям
по курсу "Основы программирования"
для студентов специальности
7.080402 "Информационные технологии проектирования"

Утверждено
редакционно-издательским
советом университета,
протокол № 1 от 30.03.07.

Харьков НТУ "ХПИ" 2007

Методические указания к лабораторным занятиям по курсу "Основы программирования" для студентов специальности 7.080402 "Информационные технологии проектирования". / Сост. С. Н. Воронцов, И. В. Костяник, В. И. Сериков. – Харьков: НТУ «ХПИ», 2007. – 52 с. – Рус. яз.

Составители: С. Н. Воронцов
И. В. Костяник
В. И. Сериков

Рецензент К. И. Богатыренко

Кафедра теории и систем автоматизированного проектирования механизмов и машин

СОДЕРЖАНИЕ

ВСТУПЛЕНИЕ

Повсеместное использование персональных компьютеров в системах автоматизированного проектирования гусеничных и колесных машин требует от современного инженера овладения навыками использования вычислительной техники. Одной из дисциплин, непосредственно связанных с применением компьютеров, является "Основы программирования". Основной раздел курса – изучение методики разработки алгоритмов и языка программирования ТурбоПаскаль для использования в создании вычислительных програм при расчетах, в курсовом и дипломном проектировании.

Паскаль – один из наиболее распространенных языков программирования 80-90-х годов, поддерживающий самые современные методологии проектирования программ (нисходящее, модульное проектирование, структурное программирование). Новую жизнь в язык вдохнула фирма Борланд, разработавшая на его базе семейство Паскаль-систем, называемых Турбо-Паскалем. Компиляторы фирмы поддерживаются многими распространенными операционными системами персональных ЭВМ, такими как CP/M-80, MSDOS, MSX DOS, среды Windows и т.п.

Интегрированная среда, обеспечивающая многооконную разработку программной системы, обширный набор встроенных в нее средств компиляции и отладки, доступный для работы через легко осваиваемое меню, – все это обеспечивает высокую производительность труда программиста.

Язык Паскаль разработан с учетом принципов структурного программирования, которое на современном этапе признано действенным методом рационализации труда программиста. Для структурированных программ характерны легкость отладки и корректировки, малая частота ошибок. Кроме этого, такие программы легко сопровождать и модифицировать без участия разработчиков.

Паскаль обладает полным набором структурных типов данных, таких, как простые переменные, массивы, файлы, множества, записи, записи с вариантами, ссылочные переменные. Введение структурных типов данных способствует созданию эффективных алгоритмов.

Особо следует отметить надежность Паскаль-программ, которая достигается иногда за счет избыточности, например, обязательного описания переменных и соответствующих типов. Надежность достигается также за счет простоты и естественности конструкций языка, соответствующих логиче-

скому мышлению разработчика программ.

Целью методических указаний является улучшение подготовки студентов специальности "Информационные технологии проектирования" в ходе изучения ими языка ТурбоПаскаль для разработки вычислительных программ, исследовании механических систем.

Методические указания состоят из восьми разделов, в которых изложен необходимый теоретический материал и приведены многочисленные примеры и задания, которые дают возможность наглядно продемонстрировать практическое применение рассматриваемой темы. Последовательность тем подчинена задачам практики.

ТЕХНИКА БЕЗОПАСНОСТИ ПРИ РАБОТЕ С ЭВМ

1. К работе на ЭВМ допускаются студенты, изучившие правила техники безопасности, имеющие навыки работы с системой и текст программы для реализации.

2. Подготовка ЭВМ к работе, техническое обслуживание и ремонт производятся персоналом кафедры, имеющим соответствующую подготовку. Поэтому о любых неполадках в работе ЭВМ необходимо сообщить преподавателю.

3. Студентам при работе на ЭВМ разрешается пользоваться только клавиатурой, манипулятором «мышь» и дисплеем. Использование других устройств без разрешения преподавателя запрещается.

4. ***Категорически запрещается*** пользоваться дискетами, не проверенными преподавателем.

1. ПРОСТЫЕ ТИПЫ ДАННЫХ

Лабораторная работа 1

Перечислимый и ограниченный тип

1.1. Цель лабораторной работы: ознакомиться с перечислимыми типами, получить навыки в организации ввода-вывода значений переменных перечислимого типа данных.

1.2. Теоретический материал

Стандартные типы данных – **integer**, **boolean, char**, **real**. Первые три типа данных являются порядковыми, т.е. к переменным этих типов применимы стандартные функции **Succ** и **Pred.**

Кроме этих типов в Паскале разрешено введение новых типов. Эти но-

вые простые типы должны быть описаны в секции типов, представленной на синтаксической диаграмме (рисунок 1.1). В соответствии со стандартом языка эта секция располагается между секцией констант и секцией переменных. Введение новых типов расширяет возможности языка Паскаль, повышает читабельность программ.

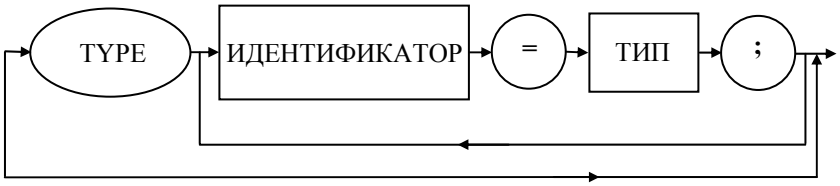


Рисунок 1.1. Секция типов

Допустим необходимо представить последовательность месяцев года. Пользуясь предопределенными типами данных, можно использовать прием, который ставит в соответствие число 1 – январю, 2 – февралю и т.д. Однако такое представление неудобно, т.к. надо трактовать числовые значения. Удобнее было бы написать **Month := MAY.**

Чтобы можно было в программе манипулировать с названиями месяцев, а не числами, в Паскале разрешено ввести новый тип. Переменные, имеющие этот тип, могут принимать значения месяцев года.

```
type
Month = (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT,
NOV, DEC) ;
var M: Month;
```

В этом случае создается новый тип данных, состоящий из последовательности значений, заключенных в круглые скобки. Такой тип данных принято называть перечислимым. Переменной перечислимого типа может быть назначено любое значение в пределах типа.

В Паскале отсутствуют средства, которые бы позволяли осуществлять непосредственный ввод-вывод переменных перечислимого типа. Если записать

```
M:=APR;
Write (M) ;
```

то слова APR выведено не будет. Поэтому выводить значения переменных перечислимого типа нужно программным путем. Однако можно вывести выражение вида **Write(Ord(M))**. При этом будет выведено число 3, соответствующее порядковому номеру идентификатора **APR** в списке значений.

Переменные перечислимого типа могут быть использованы в булевых выражениях. Кроме этого к данным этого типа применимы операции сравнения, например:

if M>MAY and M<SEP then Writeln ('Летний месяц')

Пример 1.1. Пусть перечень специальностей кафедры колесных и гусеничных машин задан следующими обозначениями: "Гусеничные и колесные машины" – GKM; "Электрические системы и комплексы транспортных средств" – ESKTS; "Информационные технологии проектирования" – ITP. Написать программу, в которой переменной перечислимого типа присваивается одно из значений GKM, ESKTS, ITP в зависимости от введенного одного символа. Если введенные символы ошибочны, то выдать соответствующее сообщение.

```
program KGM;
type spes = (GKM,ESKTS,ITP) ;
var S: spes;
   Well: boolean;
   C: char;
begin
      Read (C);
      Well := false;
      case C of
      'G': begin Well:= true; S:= GKM; end;
      'E': begin Well:=true; S:= ESKTS; end;
      'I': begin Well:=true; S:= ITP; end;
      end; { case}
      Writeln(Ord(S));
      Readln;
   if not Well then Writeln ('Ошибочный символ');
end.
```

Упорядоченность элементов перечислимого типа определяется порядком их следования. Самый левый элемент имеет минимальное значение, а наиболее правый – максимальное.

Кроме перечислимого типа в Паскале разрешено введение так называемого ограниченного или интервального типа, форма записи которого представлена на нижней ветви синтаксической диаграммы. Например: **Type**

year = 1900..2000; letter = 'A'..'Z'; spes= 1..3;.

Левая и правая константы задают диапазон значений и их называют соответственно нижней и верхней границей ограниченного типа. Значения этих констант должны удовлетворять условию: **левая константа <= правая константа**.

Пример.1.2. Определить следующую дату дня недели, если заданы текущая дата и день недели. Текущая дата включает число и номер месяца. При выводе дата отображается в виде числа и названия месяца.

```
Program PRIMER;
Type
   mounth = (jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec);
   day = 1..31;
   year = 1900..2000; Var
   d : day;
   m : mounth;
   g : year;
   num : 1..12;
Begin
   Write('==>');
   Readln(d, num, g);  { ввод даты - число, номер месяца, год }
   Case num of
      1 : m:= jan;
      2 : m:= feb;
      3 : m:= mar;
      4 : m:= apr;
      5 : m:= may;
      6 : m:= jun;
      7 : m:= jul;
      8 : m:= aug;
      9 : m:= sep;
      10 : m:= oct;
      11 : m:= nov;
      12: m:= dec;
   End;
   Case m of     { контроль на корректность количества дней в месяце}
      jan, mar, may, jul, aug, oct : If d = 31 then
                                     begin
                                        d := 1; m := succ(m)
                                     end
                                     else d := d + 1;
      apr, jun, sep, nov    :   If d = 30 then
```

```
                    begin
                      d := 1; m := succ(m)
                    end
                    else d := d + 1;
    dec          :  If d = 31 then
                    begin
                      d := 1; m := jan; g := g + 1
                     end
                    else d := d + 1;
    feb          :  If (( d = 28) and ((g mod 4 <> 0) or (g mod
                    100 = 0)) and (g mod 400 <> 0)) or (d=29)
                   then
                   begin
                      d := 1; m := succ(m);
                   end
                   else d := d + 1
  end;  { case }
  Write('Следующая дата: ', d : 2); { вывод результата }
  Case m of  { выбор названия месяца по номеру месяца}
     jan: write(' января ');
     feb: write(' февраля ');
     mar: write(' марта ');
     apr: write(' апреля ');
     may: write(' мая ');
     jun: write(' июня ');
     jul: write(' июля ');
     aug: write(' августа ');
     sep: write(' сентября ');
     oct: write(' октября ');
     nov: write(' ноября ');
     dec: write(' декабря ')
  end;  { case }
  Writeln(g : 5,' года ')
End.
```

Пояснения к программе:

В современном календаре каждый год, номер которого делится на 4, является високосным, за исключением тех номеров, которые делятся на 100 и не делятся на 400.

Введение ограниченного типа улучшает читабельность программ, так как представляет диапазон значений, которые может принимать переменная этого типа. Константы в определении ограниченного типа должны отно-

ситься к одному и тому же базовому типу (целому, символьному, порядковому). Базовый тип констант определяет допустимость соответствующих операций над данными ограниченного типа. И перечислимый, и ограниченный типы переменных относят к простому типу.

Простой – это такой тип, который может представлять только одно значение. Например, переменная типа **integer** может хранить только одно целое число.

Контрольные вопросы

1. Что такое перечислимый тип?
2. Какие операции допускаются с данными перечислимого типа?
3. Какие стандартные функции можно использовать для данных перечислимого типа?
4. Что такое ограниченный тип?
5. Может ли быть последующее значение больше предыдущего при определении ограниченного типа?
6. Какие операции допускаются над переменными ограниченного типа?

Задание к работе

Выполнить индивидуальное задание.

Методические указания

Перед выполнением индивидуального задания по вариантам 1-7, 10 ознакомьтесь с примером, приведенным выше.

При выполнении индивидуального задания придерживаться следующей технологии:

1) изучить словесную постановку задачи, выделив при этом все виды данных;

2) выбрать метод решения задачи, если это необходимо;

3) разработать программу на языке Паскаль, используя перечислимые и ограниченные типы данных;

4) разработать контрольный тест к программе;

5) отладить программу;

6) представить отчет по работе к защите.

Содержание отчета

1. Титульный лист.

2. Словесная постановка задачи.

3. Графический или текстуальный алгоритм решения задачи.

4. Листинг программы.

5. Контрольный тест и результаты тестирования программы.

6. Ответы на контрольные вопросы по согласованию с преподавателем.

Варианты индивидуальных заданий

1. Дан номер года. Указать число дней в этом году.

2. Даны натуральные числа *n, m*(*n* <= *m*). Определить, сколько из чисел *n*, *n*+1,.., *m* являются номерами високосных годов.

3. Даны натуральные числа a_1, b_1, c_1, a_2,b_2, c_2, которые указывают две даты (число, месяц, год). Вычислить количество дней прошедших между двумя этими датами.

4. Даны натуральные числа a_1, b_1, c_1, a_2, b_2, c_2, которые указывают две даты (число, месяц, год). Вычислить количество полных лет прошедших между двумя этими датами.

5. Даны натуральные числа *a*, *b*, *c* которые обозначают число, месяц и год. Проверить корректность этой даты.

6. Даны натуральные числа *a*, *b*, *c* которые обозначают число, месяц и год. Найти номер этого дня с начала года.

7. Даны натуральные числа *a*, *b*, *c* которые обозначают число, месяц и год. Определить, сколько полных дней осталось до конца года.

8. Известно, что астрологи делят год на 12 периодов, и каждый из них ставит в соответствии один из знаков зодиака:
20.1-18.2-Водолей 19.2-20.3-Рыбы 21.3-19.4-Овен
20.4-20.5-Телец 21.5-21.6-Близнецы 22.6-22.7-Рак
23.7-22.8-Лев 23.8-22.9-Дева 23.9-22.10-Весы
23.10-22.11-Скорпион 23.11-21.12-Стрелец 22.9-19.1-Козерог

Дата дана в виде dd. mm. Определить какой знак зодиака соответствует этой дате.

9. "Угадай число". Один из играющих задумывает число от 1 до 100, другой пытается угадать его за 5 вопросов вида: Верно ли, что задуманное число больше такого то числа? Написать программу, загадывающую число.

10. В некоторой библиотеке последний четверг каждого месяца – санитарный день. По заданному номеру месяца выводить на экран дату са-

нитарного дня.

2. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

Наряду с простыми типами Паскаль содержит ряд структурированных типов данных, которые могут представлять совокупности значений. В Паскале имеются следующие структурированные типы: массивы, файлы, множества, записи.

Переменные этих типов имеют структуры, которые определяются Н. Виртом, как «совокупность связанных данных и множество правил, определяющих их организацию и способ доступа к элементам данных». Выбором той или иной структуры данных мы определяем и алгоритм обработки данных, от которого зависит эффективность их обработки.

Лабораторная работа 2

Массивы

2.1. Цель лабораторной работы:

1. Освоение приемов работы с одно- и двумерными массивами в языке Паскаль.

2. Применение приемов векторной и матричной алгебры для машинной обработки данных.

3. Изучение основных алгоритмов поиска и сортировки одно- и двумерных массивов.

2.2. Теоретический материал

Массив – это упорядоченный набор переменных одного типа. Массивы содержат фиксированное число компонент, которое задается при определении переменных типа массив. Тип компонент массива – базовый. Тип индекса (индексов) в описании заключается в квадратные скобки и относится к простому.

Ограничений на количество индексов нет. Индекс задает место элемента в массиве. Тип компоненты массива может быть любым.

Примеры описания массивов:

Mas: array [1..15] of real ; (* описан массив из 15 вещественных чисел *)

Spes: array [(GKM, ESKTS, ITP)] of integer; (* описан массив целых чисел, индексы элементов массива имеют перечислимый тип и принимают значения названий специальностей GKM, ESKTS, ITP*)

B : array ['A'..'Z'] of boolean; (* описан массив элементов булевого типа, тип индексов – ограниченный символьный *)

C : array [1..3, 1..5] of real; (* описан двумерный массив с вещественных чисел, содержащий три строки и пять столбцов *)

D : array [(BLACK, WHITE)] of 11..20; (* описан массив D целых чисел с индексами BLACK, WHITE. Каждый элемент массива может принимать значения от 11 до 20*)

К первому элементу массива Mas можно обратиться Mas[l], ко второму – Mas[2] и т.д. Пример цикла, который обнуляет элементы массива Mas имеет вид:

```
for I:=1 to 15 do Mas[I]:=0;
```

Переменная **Mas [I]** называется переменной с индексом. В качестве индекса может быть любое выражение, имеющее тот же тип, что и индекс.

Пример 2.1. Написать программу для вычисления суммы элементов массива из 100 вещественных чисел.

```
Program Test;
Const N=100;
Type  Mas = array [1..N] of real;
var     Sum: real ; M: Mas;
      I: integer;
begin
   Sum:=0;
   for I:=l to N do
   Sum:=Sum+M[I];
   Writeln('Sum=',Sum);
end.
```

В Паскале многомерный массив можно описать как одномерный, элементами которого являются массивы. Упомянутую выше матрицу **C** можно описать как одномерный массив из трех элементов типа строка, каждая из которых является массивом из пяти элементов:

```
Type Mas = array [1..3] of array [1..5] of real;
Var  A, B : Mas;
```

При последовательной записи каждой строки матрицы в память соблюдается правило размещения двумерного массива по строкам. Ссылка на элемент матрицы, лежащей на пересечении *i*-той строки и *j*-oro столбца, вы-

глядит следующим образом: **A[I, J]**

Пример 2.2. Написать программу вычисления произведения двумерных вещественных матриц A, размерностью (N, M), и B, размерностью (M, L). Результирующая матрица C будет иметь размерность (N, L), причем каждый ее элемент будет вычисляться по формуле

$$c_{ij} = \sum_{k=1}^{M} a_{ik} \cdot b_{kj}, \qquad (i = 1,..., N; j = 1,..., L).$$

```
program Test;
const N=5; M=2; L=3;
var   A: array [1..N,1..M] of real ;
      B: array [1..M,1..L] of real ;
      C: array [1..N,1..L] of real ;
      i,j,k: integer;
begin {Ввод массивов A, B}
  for i:=1 to N do
      for j:=1 to L do
      begin C[i,j]:=0;
            for k:=1 to M do C[i,j]:=C[i,j]+A[i,k]*B[k,j];
            Writeln('C',i,j,'=',C[i,j]);
      end;
end.
```

Контрольные вопросы

1. Дайте определение массива.
2. Как организовать ввод матрицы размером N*M элементов?
3. Как организовать вывод матрицы?

Задание к работе

1. Выполнить индивидуальное задание А.
2. Выполнить индивидуальное задание Б.

Методические указания

При выполнении индивидуального задания необходимо соблюдать технологию решения задач на ЭВМ:

1) изучить словесную постановку задачи, выделив при этом все виды данных;

2) сформулировать математическую постановку задачи, выделив при этом все виды данных;

3) сформулировать математическую постановку задачи;

4) выбрать метод решения задачи, если это необходимо;

5) записать разработанный алгоритм на языке Паскаль;

Содержание отчета

1. Титульный лист.
2. Словесная постановка задачи.
3. Графический или текстуальный алгоритм решения задачи.
4. Листинг программы.
5. Контрольный тест и результаты тестирования программы.
6. Анализ ошибок, допущенных при программировании.
7. Ответы на контрольные вопросы по согласованию с преподавателем.

Варианты индивидуальных заданий

*Задание А***.**

Обработать на ЭВМ массив в соответствии с вариантом задания.

Вариант	Массив	Действия	Условия и ограничения
1	X(100)	Вычислить сумму и количество элементов массива X.	$0 \le x[i] \le 1$
2	A(80)	Вычислить среднее арифметическое значение элемента массива A.	$a[i] > 0$
3	X(70)	Переписать элементы массива X в массив Y и подсчитать их количество.	$-1\ x[i] \le 1$
4	B(50)	Определить максимальный элемент массива B и его порядковый номер.	$x[i] > 0$
5	C(40)	Вычислить минимальный элемент массива C и его номер.	$x[i] < 0$
6	D(80)	Найти максимальный и минимальный элементы массива D и поменять их местами.	
7	Y(20)	Вычислить среднее геометрическое элемента массива Y.	$y[i] > 0$

Продолжение

1	2	3	4
8	Z(30)	Расположить в массиве R сначала положительные, а затем отрицательные элементы массива Z.	
9	N(50)	Определить сумму элементов массива N, кратных трем.	n[i]/3*3=n[i]
10	X(N)	Вычислить сумму и количество элементов массива X.	N ≤ 40

Задание Б.

Вариант	Матрица	Действия	Условия и ограничения
1	2	3	4
1	A(10,15)	Вычислить и запомнить сумму и число положительных элементов каждого столбца матрицы. Результаты отобразить в виде двух строк.	a[i,j]>0
2	A(N,M)	Вычислить и запомнить суммы и числа элементов каждой строки матрицы. Результаты отобразить в виде двух столбцов.	N <= 20 M <= 15
3	B(N,N)	Вычислить сумму и число элементов матрицы, находящихся под главной диагональю и над ней.	N ≤12
4	C(N,N)	Вычислить сумму и число положительных элементов матрицы, находящихся над главной диагональю.	c[i,j]>0 N ≤12
5	D(K,K)	Записать на место отрицательных элементов матрицы нули и отобразить ее в общепринятом виде.	K ≤ 10
6	D(10,10)	Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы. Отобразить нижнюю треугольную матрицу в общепринятом виде.	

Продолжение

1	2	3	4
7	F(N,M)	Найти в каждой строке матрицы максимальный и минимальный элементы и поместить их на место первого и последнего элемента строки соответственно. Матрицу вывести в общепринятом виде.	N ≤20 M ≤10
8	F(10,8)	Транспонировать матрицу и вывести на печать элементы главной диагонали и диагонали, расположенной под главной.	
9	N(10,10)	Для целочисленной матрицы найти для каждой строки число элементов, кратных пяти, и наибольший из полученных результатов.	n_{ij} /5*5= n_{ij}
10	P(N,N)	Найти в каждой строке матрицы наибольший элемент и поменять его местами с элементом главной диагонали. Отпечатать полученную матрицу в общепринятом виде.	N ≤15

Лабораторная работа 3

Символьные строки. Тип String в ТурбоПаскале

3.1. Цель лабораторной работы:

1. Ознакомиться со строковыми данными.

2. Получить навыки в организации работы со строковыми переменными: удалением, вставкой, копированием, заменой одной строки на другую и т.д.

3.2. Теоретический материал

Строка – это массив, компоненты которого имеют тип **char** и тип индекса имеет нижнюю границу, равную 1.

Строки и строковые константы можно использовать в операторах присваивания, а также в процедурах Write, Writeln в качестве фактических параметров.

К строкам применимы все 6 операций отношений, но при этом строки должны иметь одинаковую длину.

В ТурбоПаскале введен тип данных **String**. Тип данных **String** иногда называют стринговым. Примеры описания стринговых переменных:

var Name: string[20]; Title: string[40]; Rez: string [70] ;

Память, отведенная для хранения значений переменной Name, составляет 21 байт. В каждом байте хранится одна литера (литера – это цифра, буква, точка или какой-нибудь другой знак). Каждая литера представляет собой значение типа **char**. Один байт переменной Name содержит ее текущую длину. Это значение не должно превышать 255. Например:

Name := 'Автор';

Title := ' Программирование на языке Паскаль ';

При присваивании значение стринговой переменной берется в кавычки (апострофы, но не парные кавычки). Переменная **Title** занимает всего 34 байта, один байт содержит ее текущую длину. В стринговых выражениях используется только одна операция – конкатенация (слияние), которая обозначается знаком "+".

Например, **Rez := Name+Title;**

Значение выражения **Rez: 'Автор Программирование на языке Паскаль'.** К стринговым переменным могут применяться следующие операции сравнения: **=, < >, >, >=, <, <=.** При этом стринговые переменные должны иметь одинаковую длину, в противном случае фиксируется ошибка.

Например:

var Age: string [3] ;...

Age := 'тридцать';...

Переменной Age будет присвоено значение «три», т.к. максимальная длина переменной Age не должна превышать трех. Лишние правые литеры усекаются.

Для переменных типа String в ТурбоПаскале допускается применение процедур Read, Readln, Write, Writeln.

Регулярную переменную типа ARRAY OF CHAR (или литерный ряд) можно рассматривать как стринг постоянной длины. Данные указанного типа могут быть использованы в любых стринговых выражениях. При этом согласование операндов по типам обеспечивается компилятором, который в подобных случаях просто преобразует литерный ряд в стринг длиной, рав-

ной количеству элементов ряда. Это позволяет, например , сравнивать литерные ряды между собой и обращаться с ними точно так же, как с переменными типа STRING. Допускается выполнять присваивание, в левой части которого стоит имя литерного ряда, в правой - стринговый литерал (константа) длиной, равной количеству элементов ряда. Однако нельзя присваивать какой-либо переменной типа литерный ряд переменную типа STRING или наоборот.

Пример 3.1.

```
Const
message = 'верно';
Type
CharArray = array[1..5] of char; Var
FixedString, FiveChar : CharArray;
VarString             : string[10];
Begin
. . .
FiveString := 'мерно';
FixedString := message;
if FiveChar > FiveString then writeln('"', FiveChar,' "больше, чем"', FiveString,'"');
VarString := 'при';
VarString := concat(VarString,FiveChar);
VarString := 'голос';
FiveChar := VarString; { так нельзя}
```

Пояснение к программе.

В рассмотренном примере объявлено два литерных ряда (типа CharArray) - FixedString и FiveChar, а также стринг VarString.

1. Стринговый литерал 'мерно' присваивается переменной FiveChar.

2. Переменной FixedString присваивается стринговая константа message, имеющая значение 'верно'.

3. Сравниваются два ряда, в результате которого выясняется, что FiveChar больше FixedString (поскольку 'мерно' > 'верно').

4. Переменная VarString получает значение 'при'.

5. В результате конкатенации VarString получает новое значение – 'примерно', которое на следующем шаге затирается значением 'голос'.

6. В последнем операторе делается попытка литерному ряду назначить стринговую переменную, это недопустимо по обычной в таких случаях ошибке "несоответствие типов". В программе левый операнд объявлен как литерный ряд, а правый как стринг.

Контрольные вопросы

1. Дайте определение строковой переменной.
2. Какие типы данных используются в качестве базовых в строковых данных?
3. Каким образом распределяется память под строковые переменные?
4. Какие операции выполняются над строковыми переменными?
5. В чем состоит сходство и различие строковых переменных и символьных массивов?
6. Возможно ли преобразование строковых переменных?
7. Назовите основные функции над строковыми переменными и их назначение.
8. Каково назначение процедур DELETE, INSERT.
9. Как реализуется ввод и вывод строковых переменных.
10. Предложите схему преобразования действительных чисел в стринг.

Задание к работе

Выполнить индивидуальное задание.

Методические указания

1. Строка символов формируется при вводе с клавиатуры.
2. При выполнении задания необходимо использовать стандартные функции и процедуры работы со стринговыми переменными: COPY, POS, LENGTH, INSERT, DELETE и операцию конкатенации.
3. Написать и отладить программу.
4. Написать отчет по работе.

Содержание отчета

1. Титульный лист.
2. Словесная постановка задачи.
3. Графический или текстуальный алгоритм решения задачи.
4. Листинг программы.
5. Контрольный тест и результаты тестирования программы.

6. Анализ ошибок, допущенных при программировании.

7. Ответы на контрольные вопросы по согласованию с преподавателем.

Варианты индивидуальных заданий

1. Дана строка символов. Подсчитать сколько среди символов данной строки встречается буква x.

2. Дана строка символов. Подсчитать:

а) сколько раз в данной строке встречается символ + и сколько раз символ *;

б) общее число вхождений символов +, -, * в строку.

3. Дана строка символов. Преобразовать данную строку, заменив в ней:

а) все восклицательные знаки точками;

б) каждую точку многоточием.

4. Дана строка символов S. Выяснить имеются ли в данной строке такие символы s_i, s_{i+1}, что s_i - это запятая, а s_{i+1} - это тире.

5. Даны две строки символов S1 и S2. Выяснить, верно ли, что среди символов строки S1 имеются все буквы строки S2.

6. Дана строка символов. Удалить из данной строки все группы букв вида *asdf*.

7. Дана строка символов. Преобразовать строку, удалив каждый символ * и повторив каждый символ, отличный от *.

8. Дана строка символов. Заменить в данной строке каждую группу букв child группой букв children.

9. Дана строка символов. Исключить из строки группы символов, расположенные между парами скобок (,), {, }. Сами скобки тоже должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.

10. Дана строка символов. Словом будем называть группы символов, разделенных пробелами (одним или несколькими). Подсчитать количество слов в данной строке.

Лабораторная работа 4

Файлы

4.1. Цель лабораторной работы:

1. Изучение файловых типов в языке Турбо-Паскаль.

2. Получение навыков в организации файлов и использовании их для

обработки информации.

4.2. Теоретический материал.

Слово «файл» в Паскале употребляется для объектов, состоящих из последовательности компонент одного типа. Число компонент, называемое длиной файла, в определении файлового типа не фиксируется.

Например:

```
type
Text = file of char;
Int = file of integer;
```

Тип **Text** называется текстовым и является стандартным. Стандартный файловый тип Text определяет файл, содержащий символы, упорядоченные в строки. Текстовые файлы используют специальные процедуры ввода-вывода.

Компоненты файла могут иметь любой тип за исключением файлового, т.е. в данном случае недопустима рекурсия по описанию вида:

```
type
A =file of file of...
```

Если слово **of** и тип компонента в описании опущены, то тип обозначает нетипизированный файл. Нетипизированные файлы представляют собой каналы ввода-вывода нижнего уровня, в основном используемые для прямого доступа к любому файлу на диске, независимо от его внутреннего формата.

Основой ввода-вывода в Паскале является последовательный файл. Над последовательным файлом определены два вида действий. Этот файл можно просматривать (читать) по одному элементу с начала в каждый момент времени. В этот файл можно записывать новые элементы, путем добавления их в конец файла (в начале записи этот файл пустой). Файлы, которые используются для ввода(вывода) в Паскаль-программе, существуют до (или после) выполнения программы. Их принято называть «внешними». В Паскале имеются три класса файлов: типизированный файл, текстовый файл и нетипизированный файл.

Файловый ввод-вывод. Перед использованием файловой переменной она должна быть связана с внешним файлом с помощью вызова процедуры Assign. Внешним файлом обычно является поименованный файл на диске, но он также может представлять собой устройство, например, кла-

виатуру или дисплей. Во внешних файлах сохраняется записанная в файл информация, или они служат источниками информации, которая считывается из файла.

Когда связь с внешним файлом установлена, для подготовки ее к операции ввода или вывода файловая переменная должна быть "открыта". Существующий файл можно открыть с помощью процедуры Reset, а новый файл можно создать и открыть с помощью процедуры Rewrite. Текстовые файлы, открытые с помощью процедуры Reset доступны только по чтению, а текстовые файлы, открытые с помощью процедуры Rewrite, доступны только по записи. Типизированные и нетипизированные файлы всегда допускают как чтение, так и запись, независимо от того были они открыты с помощью процедуры Reset или с помощью процедуры Rewrite.

Любой файл, представляет собой линейную последовательность элементов, каждый из которых имеет тип элемента (или тип записи) файла. Каждый элемент файла имеет номер. Первый элемент файла считается нулевым элементом.

Обычно доступ к файлам организуется последовательно, то есть, когда элемент считывается с помощью стандартной процедуры Read или записывается с помощью стандартной процедуры Write, текущая позиция файла перемещается к следующему по порядку элементу файла. Однако к типизированным и нетипизированным файлам можно организовать прямой доступ с помощью стандартной процедуры Seek, которая перемещает текущую позицию файла к заданному элементу. Для определения текущей позиции в файле и текущего размера файла можно использовать стандартные функции FilePos и Filesize.

Когда программа завершает обработку файла, он должен закрываться с помощью стандартной процедуры Close. После полного закрытия файла связанный с ним внешний файл обновляется. Затем файловая переменная может быть связана с другим внешним файлом.

По умолчанию при всех обращениях к стандартным функциям и процедурам ввода-вывода автоматически производится проверка на наличие ошибок. При обнаружении ошибки программа прекращает работу и выводит на экран сообщение об ошибке. С помощью директив компилятора {$I+} и {$I-} эту автоматическую проверку можно включить или выключить. Когда автоматическая проверка отключена, то есть когда процедура

или функция была скомпилирована с директивой {$I-}, ошибки ввода-вывода, возникающие при работе программы, не приводят к ее останову. При этом, чтобы проверить результат выполнения операции ввода-вывода, нужно использовать стандартную функцию IOResult.

Для очистки ошибки, которая может произойти, вы можете вызвать функцию IOResult. Если вы этого не сделаете, и текущим состоянием является {$I+}, то из-за оставшейся ошибки IOResult следующая операция ввода-вывода завершится с ошибкой.

Примечание: Если вы пишете программу дл Windows и не хотите, чтобы Windows обрабатывала за вас ошибки ввода-вывода на диск или другие ошибки ввода-вывода, вызовите SetErrorMode(1).

Текстовые файлы. Опишем операции ввода и вывода, использующие файловую переменную стандартного текстового типа. Заметим, что в Borland Pascal текстовый тип (тип Text) отличается от символьного типа Char.

При открытии текстового файла внешний файл интерпретируется особым образом: считается, что он представляет собой последовательность символов, сгруппированных в строки, где каждая строка заканчивается символом конца строки (end-of-line), который представляет собой символ перевода каретки, за которым возможно следует символ перевода строки.

Для текстовых файлов существует специальный вид операций чтения и записи (read и write), который позволяют вам считывать и записывать значения, тип которых отличается от символьного типа Char. Такие значения автоматически переводятся в символьное представление и обратно. Например, Read(f,i), где i – переменная целого типа, приведет к считыванию последовательности цифр, интерпретации этой последовательности, как десятичного числа, и сохранению его в i.

Как было отмечено ранее, имеются две стандартных переменных текстового типа – это Input и Output. Стандартная файловая переменная Input – это доступный только по чтению файл, связанный со стандартным файлом ввода операционной системы (обычно это клавиатура), а стандартная файловая переменная Output – это доступный только по записи файл, связанный со стандартным файлом вывода операционной системы (обычно это дисплей). Перед началом выполнения программы DOS файлы Input и Output

автоматически открываются, как если бы были выполнены следующие операторы:

Assign(Input,''); Reset(Input); Assign(Output,''); Rewrite(Output);

Так как Windows не поддерживает непосредственно ориентированный на текст ввод и вывод, файлы Input и Output по умолчанию в прикладной программе Windows не присваиваются, и любая попытка чтения из этих файлов или записи в них приведет к ошибке ввода-вывода. Однако, если прикладная программа использует модуль WinCrt, то Input и Output будут ссылаться на прокручиваемое текстовое окно. Модуль WinCrt содержит всю логику управления, необходимую для эмуляции текстового экрана в операционной среде Windows, поэтому в прикладной программе, использующей модуль WinCrt, не требуется никаких приемов программирования, специфических для Windows.

Для некоторых из стандартных процедур и функций, список которых приведен в данном разделе, не требуется явно указывать в качестве параметра файловую переменную. Если этот параметр опущен, то по умолчанию будут рассматриваться переменные Input или Output, в зависимости от того, будет ли процедура или функция ориентирована на ввод или на вывод. Например, Read(x) соответствует Read(Input,x) и Write(x) соответствует Write(Output,x).

Если при вызове одной из процедур или функций из этого раздела вы задаете файл, этот файл должен быть связан с внешним файлом с помощью процедуры Assign и открыт с помощью процедуры Reset, Rewrite или Append. Если для ориентированной на вывод процедуры или функции вы указываете файл, который был открыт с помощью процедуры Reset, то выведется сообщение об ошибке. Аналогично, будет ошибкой задавать для ориентированной на ввод процедуры или функции файл, открытый с помощью процедур Rewrite или Append.

Нетипизированные файлы. Нетипизированные файлы представляют собой каналы ввода-вывода нижнего уровня, используемые в основном для прямого доступа к любому файлу на диске, независимо от его типа и структуры. Любой нетипизированный файл описывается словом **file** без атрибутов. Например:

```
var     DataFile: file;
```

Для нетипизированных файлов в процедурах Reset и Rewrite допускается указывать дополнительный параметр, чтобы задать размер записи, использующийся при передаче файла.

По историческим причинам принимаемая по умолчанию длина записи равна 128 байтам. Предпочтительной длиной записи является длина записи, равная 1, поскольку это единственное значение, которое позволяет точно отразить размер любого файла (когда длина записи равна 1, то в файле не могут присутствовать неполные записи, то есть записи с меньшей длиной).

За исключением процедур Read и Write для всех нетипизированных файлов допускается использование любой стандартной процедуры, которые допускается использовать с типизированными файлами. Вместо процедур Read и Write здесь используются соответственно процедуры Blockread и BlockWrite позволяющие пересылать данные с высокой скоростью.

Переменная FileMode. Переменная FileMode, определенная в модуле System, задает код доступа, передаваемый в DOS для типизированных и нетипизированных файлов (не для текстовых файлов), когда они открываются с помощью процедуры Reset.

По умолчанию значение FileMode = 2. При этом допускается чтение и запись файла. Присваивание FileMode другого значения приводит к использованию этого режима при всех последующих вызовах Reset.

Примечание: Новые файлы, открываемые с помощью Rewrite, всегда открываются в режиме чтения/записи, что соответствует Filemode = 2.

Диапазон допустимых значений FileMode зависит от используемой версии DOS. Однако во всех версиях определены следующие режимы:

0: доступ только по чтению

1: Только запись

2: Чтение/запись

В DOS версии 3.x определены дополнительные режимы, которые касаются в основном совместного использования файлов при работе в сети (более подробно это описывается в "Руководстве программиста по DOS").

Пример 4.1. Программа для процедуры Assign.

```
Program TM;
Var F : Text;

Begin
```

```
  Assign(F, '');  { Стандартное устройство вывода }
  ReWrite(F);
  WriteLn(F, 'Стандартное устройство вывода...');
  Close(F);
End.
```

Пример 4.2. Программа для процедуры ReWrite.

```
Program TM;
Var F : Text;

 Begin
   Assign(F, 'NEWFILE.$$$');
   ReWrite(F);
   WriteLn(F, 'Только что созданный файл с этим текстом внутри...');
   Close(F);
 End.
```

Контрольные вопросы

1. Укажите режимы ввода информации.
2. В каких случаях удобно использовать файлы?
3. Дайте определение файла и укажите его характеристики.
4. Что такое путь доступа к файлу?
5. Где хранятся файлы?
6. Выведите формулу подсчета объема файла в байтах.
7. Каким образом описываются переменные файловых типов?
8. Как подразделяются файлы по видам доступа к его компонентам ?
9. Как осуществляется доступ к компонентам файлов?
10. Какие операции определены над файлами?

Задание к работе

Задание А. Разработать программу в соответствии с вариантом задания, которая должна выполнять следующие функции:

- создание файла;
- чтение данных из файла;
- вывод считанных данных на экран дисплея.

Задание Б. В программу, разработанную по заданию А, добавить блок обработки данных, инцидентных файлу, в соответствии с индивидуальным заданием. Все полученные результаты отобразить на экране.

Методические указания

1. При разработке процедуры создания файла необходимо придерживаться следующей схемы действий:

1) проверить с помощью процедуры DISKSIZE, есть ли место на диске;

2) проверить, нет ли файла с таким же DOS – именем на диске (процедура FINDFIRST, FINDNEXT);

3) привести в соответствие DOS-ое имя файла с файловой переменной, используемой в программе (процедура ASSIGN);

4) открыть файл (процедура REWRITE);

5) ввести данные, предназначенные для записи в файл;

6) записать данные в файл (предложения WRITE / WRITELN);

7) закрыть файл (процедура CLOSE).

2. При создании процедуры чтения необходимо:

1) проверить, существует ли такой файл на диске (процедура FINDFIRST, FINDNEXT);

2) если файл не существует, то необходимо уточнить имя в интерактивном режиме и снова перейти к пункту а);

3) если файл существует, привести в соответствие DOS-ое имя файла с файловой переменной, используемой в программе (процедура ASSIGN);

4) открыть файл (процедура RESET);

5) считать данные из файла (предложения READ / READLN);

6) отобразить считанные данные на экране дисплея;

7) закрыть файл (процедура CLOSE).

3. В начале каждой процедуры необходимо:

1) отключить стандартную проверку выполнения операций ввода-вывода, используя директиву компилятора {$I-};

2) после выполнения каждой операции ввода-вывода самостоятельно проверять код ее завершения с помощью функции IORESULT;

3) при неуспешном завершении операции ввода-вывода устранить причину, приведшую к этой ситуации.

Содержание отчета

1. Титульный лист.

2. Словесная постановка задачи.

3. Математическая постановка задачи.

4. Таблица идентификаторов входных и выходных данных, а также их типов.

5. Листинг программы.

6. Контрольный тест.

7. Результаты тестирования.

8. Анализ допущенных ошибок.

9. Инструкция по эксплуатации программы.

10. Ответы на контрольные вопросы.

Варианты индивидуальных заданий

Вариант 1.

А. Создать файл, содержащий сведения о месячной заработной плате рабочих завода. Каждая запись содержит поля – фамилия рабочего, наименование цеха, размер заработной платы за месяц. Количество записей - произвольное.

Б. Вычислить общую сумму выплат за месяц по цеху X, а также среднемесячный заработок рабочего этого цеха. Вывести ведомость для начисления заработной платы рабочим этого цеха.

Вариант 2.

А. Создать файл, содержащий сведения о количестве изделий, собранных сборщиками цеха за неделю. Каждая запись содержит поля – фамилия сборщика, количество изделий, собранных им ежедневно в течение шестидневной недели (в понедельник, вторник и т.д.). Количество записей – произвольное.

Б. По каждому сборщику просуммировать количество деталей, собранное им за неделю. Определить сборщика, собравшего наибольшее число изделий, и день, когда он достиг наивысшей производительности труда.

Вариант 3.

А. Создать файл, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочим за месяц. Структура записи имеет поля – фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Количество записей – произвольное.

Б. Считая заданными значения расценок Sa, Sb, Sc за выполненную работу по сборке единицы изделия категорий A, B, C соответственно, подсчитать:

- общее количество изделий категорий A, B, C, собранных рабочим цеха X;
- ведомость заработной платы рабочих цеха X;
- средний размер заработной платы работников этого цеха.

Вариант 4.

А. Создать файл, содержащий сведения о телефонах абонентов. Каждая запись имеет поля – фамилия абонента, год установки телефона, номер телефона. Количество записей – произвольное.

Б. По вводимой фамилии абонента выдать номер телефона. Определить количество установленных телефонов с XXXX года. Номер года вводится с терминала.

Вариант 5.

А. Создать файл, содержащий сведения об ассортименте игрушек в магазине. Структура записи – название игрушки, цена, количество, возрастные границы, например 2÷5, т.е. от двух до пяти лет. Количество записей – произвольное.

Б. Найти игрушки, которые подходят детям от 1 до 3 лет. Определить стоимость самой дорогой игрушки и ее наименование. Определить игрушку, которая по стоимости не превышает X руб. и подходит ребенку в возрасте от A до B лет. Значения X, A, B ввести с терминала.

Вариант 6.

А. Создать файл, содержащий сведения о сдаче студентами первого курса сессии. Структура записи – индекс группы, фамилия студента, оценки по пяти экзаменам, признак участия в общественной работе: "1" – активное участие, "0" – неучастие. Количество записей – 30.

Б. Зачислить студентов группы X на стипендию. Студент, получивший все оценки "5" и активно участвующий в общественной работе, зачисляется на повышенную стипендию (доплата 50 %), не активно участвует – доплата 25 %. Студенты, получившие "4" и "5" , зачисляются на обычную стипендию. Студент, получивший одну оценку "3", но активно занимающийся общественной работой, также зачисляется на стипендию, в противном случае зачисление не производится. Индекс группы вводится с терминала.

Вариант 7.

А. Создать файл, содержащий сведения о сдаче студентами сессии. Структура записи – индекс группы, фамилия студента, оценки по пяти экзаменам и пяти зачетам ("З" означает зачет, "Н" – незачет). Количество записей – 25.

Б. Определить фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей. Найти средний балл, полученный каждым студентом группы X, и всей группой в целом.

Вариант 8.

А. Создать файл, содержащий сведения о личной коллекции книголюба. Структура записи – шифр книги, автор, название, год издания, местоположение (номер стеллажа, шкафа и т.д.). Количество записей – произвольное.

Б. Найти:

1) местонахождение книги автора X названия Y;

2) список книг автора Z, находящихся в коллекции;

3) число книг издания XX года, имеющееся в библиотеке. Значения X, Y, Z, XX ввести с терминала;

Вариант 9

А. Создать файл, содержащий сведения о наличии билетов и рейсах Аэрофлота. Структура записи – номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Количество записей – произвольное.

Б. Найти время отправления самолетов в город X; наличие свободных мест на рейс в город X с временем отправления Y. Значения X, Y вводятся по запросу с терминала.

Вариант 10.

А. Создать файл, содержащий сведения об ассортименте обуви в магазине фирмы. Структура записи – артикул, наименование, количество, стоимость одной пары. Количество записей – произвольное. Артикул начинается с буквы Д для дамской обуви, М для мужской, Р для детской.

Б. Определить наличие в файле обуви артикула X, узнать ее стоимость; ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели. Значение X вводится по запросу с терминала.

Лабораторная работа 5

Множества

5.1. Цель лабораторной работы:

1. Получение навыков в задании переменных типа множество и организации ввода и вывода данных типа множество.

2. Получение практических навыков в выполнении операций над множествами.

5.2. Теоретический материал

Паскаль разрешает оперировать с множествами, как с типами данных. Например, описание вида:

```
type
CharSet = setof'A'..'Z'
```

определяет множественный тип, значениями которого являются множества символов-букв, а членами множества – символы-латинские буквы от A до Z.

Описание вида:

```
type
NumberSet = set of 0..50
```

определяет множественный тип, значениями которого являются множества целых чисел, а членами множества – целые числа, лежащие в пределах от 0 до 50.

Пустое множество является элементом всех типов множеств.

Примеры описаний множественных типов:

```
type
SymbolSet = set of ' '..' ';
Colour= (WHITE, BLUE, RED) ;
ColourSet=set of Colour ;
T1 = set of 0..9 ; var
C : colour; CoISet: ColourSet;
T: integer;
TSet: T1;
```

В данном случае значением переменной T может быть любая цифра от 0 до 9, а значением переменной TSet – произвольная совокупность цифр от 0 до 9.

Над множествами в Паскале допустимы четыре операции:

объединение (" + ");

пересечение (" * ");

разность (" – ");

операция **in**.

Операция **in** позволяет определить, принадлежит элемент множеству или нет. Первым операндом, расположенным слева от слова in, является выражение базового типа (т.е. типа, которому должны принадлежать все члены множества). Второй операнд, стоящий справа от слова **in,** должен иметь множественный тип.

Например:

RED in [RED, WHITE] – результат **true**

8 in [0..3, 6, 9] – результат false

В Паскаль-программе множество задается в виде списка элементов, заключенного в квадратные скобки. В скобках может быть один или более элементов, а может не быть ни одного (пустое множество). В качестве элемента может использоваться константа, переменная, выражение, значение которого принадлежит базовому типу, а также паре элементов, разделенных двумя точками (интервал значений).

В Паскале можно использовать инструкции присваивания следующих видов:
ColSet := [WHITE, RED] ; ColSet := [] ; TSet := [1,7,5] ; TSet:=[1..5,8]; Tset := **[8 mod 4,15 div 5]**

При работе с множествами можно использовать операции сравнения =, <>, <=, >=.

Операции «=» и «<>» позволяют проверить, равны два множества или нет. С помощью операций «>=» и «<=» можно определить, является ли одно множество подмножеством другого.

Примеры:

[RED, WHITE]=[RED, GREEN] результат false

[RED]<=[RED, WHITE] результат true

[l]<=[0..5] результат true

Расположим операции над множествами в порядке убывания приоритета:

*****

+

in, =,<>, >=, <=

В каждой строке находятся равноприоритетные операции.

Пример 5.1. Из файла **Input** вводится текст, содержащий символы от знака «+» до левой квадратной скобки «[». Распечатать символы текста в порядке кода ASCII (из повторно встречающихся символов выводится только один).

```
Program Sort;
var
   S: char;
   Sets: set of  '+'..'[';
   I:'+'..'[';

begin
  Sets := [] ;
  Read (S);
  while not Eof do begin
       while not Eoln do begin
       Sets := Sets + [S];
       Read (S)
     end;
     Readln
  end;
     For I:='+' to '[' do
       if I in Sets then Write (I);
       writeln
end.
```

Контрольные вопросы

1. Что понимается под множеством?
2. Какие вы знаете операции над множествами в математике?
3. Как записываются операции над множествами в языке Турбо-Паскаль?
4. Как задаются множества на языке Турбо-Паскаль?
5. Что такое пустое множество и как оно задается?
6. Как организовать вывод элементов множества?

Задание к работе

1. Выполнить задание А.

2. Выполнить задание Б.

Методические указания

1. При выполнении индивидуального задания А необходимо:

1) ознакомиться с конечным и упорядоченным множеством символов, определенным на используемой для выполнения задания ЭВМ;

2) составить программу для конкретного варианта, работающую для произвольного набора символов.

3) входная строка символов может быть длиннее строки экрана терминала, при этом программа работает не с функцией EOLN, а с признаком конца строки, который задается программистом.

2. При выполнении индивидуального задания Б необходимо учесть приемы программирования, использованные в приведенной ниже программе ASMAG.

Известен набор продуктов – хлеб, масло, сыр, молоко, имеющихся в ассортименте магазинов. В три магазина доставлены отдельные виды этих продуктов. Требуется построить множества A, B, C, которые содержат соответственно:

- продукты, имеющиеся одновременно во всех магазинах;
- продукты, имеющиеся по крайней мере в одном из магазинов;
- продукты, которых нет ни в одном из магазинов.

Например**:**

```
Program ASMAG;
  Const N=3;
  Type
    product=(bread,butter,cheese,milk); {задается список объектов (продуктов), определяющий базовый тип PRODUCT}
    assort = set of product;  {на базовом типе PRODUCT определяется множественный тип ASSORT}
    magazin = array [1..N] of assort;{информация о наличии продуктов во всех магазинах задается как массив множеств}
  Var
   m1 : magazin; x : product; a,b,c, xm1 : assort; i,j,iw,m : integer;

Begin
```

```
for i := 1 to N do     {ввод исходной информации}
begin
   xm1 := [];
   writeln (' введите номера продуктов',i : '-го магазина =');
   repeat      {в цикле REPEAT формируется множество XM1, характе-
               ризующее наличие товаров в одном магазине.}
      read(iw);
      case iw of
        1:   x := bread;
        2:  x := butter;
        3:  x := cheese;
        4:  x := milk
      end;
      xm1 := xm1 + [x];
   until eoln;
   m1[i] := xm; {информация о наличии товаров записывается в мас-
   сив M1}
end;
for i := 1 to 3 do {формирование множеств A,B,C и их распечатка}
begin
   case i of
      1: writeln('продукты, имеющиеся одновременно во
      всех магазинах');
      2: writeln('ассортимент продуктов');
      3: writeln('продукты, которых нет ни в одном магазине')
   end;
   for x := bread to milk do
   if x IN a then
      case x of
         bread: write('хлеб');
         butter    : write('масло');
         cheese    : write('сыр');
         milk   : write('молоко')
      end;
if i = 1 then a := b else a := c;
```

```
    writeln
    end
end.
```

Содержание отчета

1. Титульный лист.
2. Словесная постановка задачи.
3. Графический или текстуальный алгоритм решения задачи.
4. Листинг программы.
5. Контрольный тест и результаты тестирования программы.
6. Инструкция по эксплуатации программы.
7. Ответы на контрольные вопросы.

Варианты индивидуальных заданий

Задание А.

Дана непустая последовательность символов. Требуется построить и напечатать множество, элементами которого являются символы, встречающиеся в последовательности индивидуального варианта:

Вариант	Последовательность символов
1	Буквы от 'A' до 'F' и от 'X' до 'Z';
2	Цифры от '5' до '9' и знаки арифметических операций;
4	Буквы от 'T до 'X' и цифры от '1' до '4';
5	Знаки препинания и операций отношения;
6	Знаки арифметических операций и буквы от 'E' до 'N';
7	Буквы от 'A' до 'Z' и знаки препинания;
8	Знаки операций отношения;
9	Цифры от '0' до '9';
10	Знаки арифметических операций и операций отношения.

Задание Б.

1. Даны два конечных множества A и B, элементами которых могут быть любые целые числа в диапазоне от 1 до 30. Найти прямое произведение этих множеств и вывести его на экран.

2. Даны два прямоугольника. Множества A и B – это множества точек, принадлежащих соответствующим прямоугольникам. Координаты точек - это натуральные числа от 1 до 10. Определить пересекаются ли данные прямоугольники, если пересекаются, то вывести на экран их общие точки.

3. Даны два конечных множества X и У, состоящие из целых чисел. Определить выполняется ли равенство: (A∩B)\B=A.

4. Даны два конечных множества X и У, состоящие из целых чисел. Определить выполняется ли равенство: (A\B) ∩ (B\A)=(A∩B) \ (A∪B).

5. Пусть A, B, C - конечные множества, такие что B R A R C. Найдите множество X, удовлетворяющее условиям A ∪ X =B и A ∩ X =C.

6. Пусть A, B, C - конечные множества, такие что B R A, A ∪ C= 0. Найдите множество X, удовлетворяющее условиям A \ X=B и X \ A =C.

7. Даны следующие множества A={1, 2, 3}, B={2, 3, 5, 4}, U={0, 1, 2, 3, .. ,9 }. Найти и вывести на экран A ∩ B, , A \ B, B \ A, U \ A.

8. Даны два конечных множества A и B, состоящие из целых чисел. Найти и вывести на экран (A∩B) \ (A∪B).

9. Даны два множества A={1, 2}, B={3, 4, 5}. Выведите на экран элементы множеств A x B, BxA.

10. Пусть A={b, o}. Перечислите элементы множеств A^3 и A^4.

Лабораторная работа 6

Записи

6.1. Цель лабораторной работы:

1. Получить навыки в организации ввода и вывода значений комбинированных типов данных.

2. Получить навыки программирования задач с использованием записей.

6.2. Теоретический материал

Запись – наиболее общий и гибкий структурированный тип в Паскале. Запись состоит из фиксированного числа компонент, называемых полями, которые могут быть различных типов. Этим запись существенно отличается от массива, все компоненты которого должны быть одного и того же типа.

Например,

```
type Date = record
      Year: integer;
      Month : 1 ..12 ;
      Day: 1..31
     end;
```

```
{Тип Date включает три поля: Year, Month, Day.}
type Book = record
        Title: string [40] ;
        Author: string [50] ;
        Entry: Date
      end;
var Dl: Date;
    b: Book;
```

Объявление типа запись осуществляется с помощью ключевого слова **record**, за которым следует список полей записи. Завершается описание этой структуры ключевым словом **end**. Для каждого поля должны быть указаны имя и тип. Именем поля может быть любой идентификатор.

Тип **Book** содержит три поля, одно из которых имеет ранее определенный тип **Date.**

В Паскале разрешено использовать массивы записей.

Чтобы обратиться к отдельной компоненте записи, необходимо задать имя записи, за ним точку и сразу за точкой написать название нужного поля, например,

```
Dl.day := 25;
B.title := 'computer games';
B.author := 'Levy';
```

Пример 6.1 . В памяти находится массив из 1000 элементов, каждый из которых имеет тип записи Book (см. выше). Написать Паскаль-программу определения количества книг, год издания которых меньше или равен 1600. Для каждого из таких элементов массива распечатать название книги, имя автора, год издания.

```
Program Rarity;
type
  Book = record
    Title: string[40] ;
    Author: string[50] ;
    Entry : Date ;
  end;
  Date = record
    Year: integer;
```

```
      Month: 1..12;
      Day: 1..31
   end;
var
   S, I: integer;
   Mas : array [1..1000] of Book;

Begin
   S:=0;
   For I:=l to l000 do
    ifMas[I].entry<=1600
      then begin
         Writeln (Mas[I].title,7, Mas[I].author,7, Mas[I].entry.date:6);
         S:=S+1
      end
end.
```

В Паскале предусмотрено средства, позволяющие сократить утомительные обозначения элементов записи. Форма записи оператора:

with <имя записи> **do** <оператор>

Оператор может быть составным. Оператор **with** открывает область действия, содержащую имена полей указанной переменной типа записи, так что эти имена фигурируют в качестве имен переменных (имя идентификатора записи перед ними не указываются). Кроме экономии места при написании программы, оператор with экономит также время при выполнении программы, так как ссылка на запись подготавливается только один раз.

Например,

```
With Dl, B do
   begin { with }
      Day := 25;
      Title := 'Computer Games';
      Author := 'Levy'
   end { with }
```

Контрольные вопросы

1. Что понимается под записью в языке Паскаль?
2. Как объявляются записи?

3. Какие операции допустимы над полями записи?
4. Как организовать ввод и вывод данных типа записи?
5. Как осуществляется доступ к полям записи?
6. Можно ли использовать в записи поля одного типа?
7. Чем отличается запись от массива?
8. Каково назначение оператора присоединения?

Задание к работе

Выполнить индивидуальное задание.

Методические указания

1. При выполнении работы использовать массив записей.
2. Разработать алгоритмы и программы для решения задач заданий.
3. Скомпилировать программы.
4. Составить контрольные тесты и протестировать программы.
5. Составить отчет и представить его к защите.

Содержание отчета

1. Титульный лист.
2. Словесная постановка задачи.
3. Графический или текстуальный алгоритм решения задачи.
4. Листинг программы.
5. Контрольный тест и результаты тестирования программы.
6. Инструкция по эксплуатации программы.
7. Ответы на контрольные вопросы.

Варианты индивидуальных заданий

1. Дан список учебной группы, включающий 20 человек. Для каждого студента известны: фамилия, имя, дата рождения, оценки по всем дисциплинам за последний семестр. Составить программу, которая обеспечивает ввод информации и отображение ее на экран в виде таблицы. Отобразить на экран анкетные данные студентов-отличников в виде таблицы. Отобразить на экран фамилию и имя студентов, родившихся зимой и весной.

2. Сведения об экзамене содержат следующие данные: дисциплину (программирование, алгебра, история, геометрия), дату сдачи экзамена (год, месяц, день), сведения о студенте (факультет, курс, группа, номер в журнале) и экзаменационную оценку. Задан набор сведений об экзаменах, сдан-

ных студентами за последние два года; в них факультет и предмет кодируются первыми буквами названия. Определить количество неуспевающих по программированию на экономическом факультете среди студентов первого курса, сдававших экзамены зимой 1995 года, вывести на экран их группу и номер в журнале.

3. Сведения об экзамене содержат следующие данные: дисциплину (программирование, социология, иностранный язык, физика), дату сдачи экзамена (год, месяц, день), сведения о студенте (фамилия, факультет, курс, группа) и экзаменационную оценку. Задан набор сведений об экзаменах, сданных студентами за последние несколько лет; в них факультет и предмет кодируются первыми буквами названия. Определить количество отличников по программированию на технологическом факультете среди студентов первого курса, сдававших экзамены летом 1995 года, вывести на экран их фамилии и группу.

4. Сведения об экзамене содержат следующие данные: дисциплину (программирование, вычислительная техника, информатика), дату сдачи экзамена (год, месяц, день), сведения о студенте (факультет, курс, группа, номер в журнале) и экзаменационную оценку. Задан набор сведений об экзаменах, сданных студентами за последние несколько лет; в них факультет и предмет кодируются первыми буквами названия. Определить, на каком факультете самый высокий средний балл по программированию среди студентов первого и второго курсов, сдававших экзамены зимой 1995 года.

5. Сведения об экзамене содержат следующие данные: дисциплину (программирование, вычислительная техника, информатика), дату сдачи экзамена (год, месяц, день), сведения о студенте (факультет, курс, группа, номер в журнале) и экзаменационную оценку. Задан набор сведений об экзаменах, сданных студентами за последние несколько лет; в них факультет и предмет кодируются первыми буквами названия. Определить, на каком факультете самый высокий показатель качества успеваемости по информатике (то есть самый высокий процент отличников и хорошистов) среди студентов первого курса, сдававших экзамены зимой 1995 года или летом 1996 года.

6. Справка о междугороднем телефонном разговоре содержит: номер телефона абонента (6 цифр), дату (год, месяц, день), время (час, минута), код города (3 цифры), номер телефона в другом городе (7 цифр), продолжительность разговора (в минутах), категорию (срочный, обычный) и тариф

(плата в рублях за минуту). Определить дату такого телефонного разговора, которой является максимальным по продолжительности среди срочных разговоров за указанный месяц.

7. Справка о междугороднем телефонном разговоре содержит: номер телефона абонента (6 цифр), дату (год, месяц, день), время (час, минута), код города (3 цифры), номер телефона в другом городе (7 цифр), продолжительность разговора (в минутах), категорию (срочный, обычный) и тариф (плата в рублях за минуту). Вывести на экран код города и номер телефона в другом городе для телефонных разговоров, состоявшихся с телефона 235678 8 марта 1996 года.

8. Справка о междугороднем телефонном разговоре содержит: номер телефона абонента (6 цифр), дату (год, месяц, день), время (час, минута), код города (3 цифры), номер телефона в другом городе (7 цифр), продолжительность разговора (в минутах), категорию (срочный, обычный) и тариф (плата в рублях за минуту). Вывести на экран номер телефона абонента, код города и номер телефона в другом городе для срочных телефонных разговоров, состоявшихся между 15 марта и 12 апреля 1996 года.

9. Деталь автомобиля описывается инвентарным номером (положительное целое число), весом (в килограммах), ценой и стоимостью (в рублях), датой начала производства (год, месяц, день), статусом (имеет или не имеет знак качества) и объемом производства (в штуках за смену). В заданной последовательности сведений о деталях найти инвентарные номера деталей с наибольшей датой начала производства среди всех заданных деталей. Вывести на экран инвентарный номер, объем производства, цену и стоимость деталей со знаком качества.

10. Деталь автомобиля описывается инвентарным номером (положительное целое число), весом (в килограммах), ценой и стоимостью (в рублях), датой начала производства (год, месяц, день), статусом (имеет или не имеет знак качества) и объемом производства (в штуках за смену). В заданной последовательности сведений о деталях найти инвентарные номера деталей с минимальным весом среди деталей без знака качества. Вывести на экран инвентарный номер, объем производства, цену и стоимость деталей, выпускаемых с февраля 1977 года.

Лабораторная работа 7

Указатели

7.1. Цель лабораторной работы:

1. Ознакомиться с простой динамической структурой данных – однонаправленным списком.

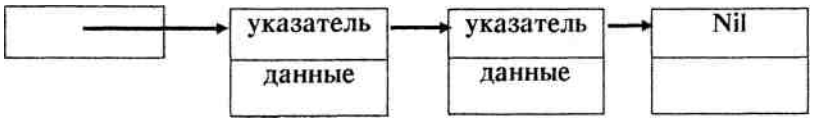
2. Получить навыки работы с переменными ссылочного типа.

3. Получить навыки программирования списков и операций над ними.

7.2. Теоретический материал

Существует ряд задач, для которых невозможно предсказать размер памяти в момент написания программы. В этих задачах необходим другой подход: переменные нужно формировать динамически, не связывая их со структурой программы. Так как средств для явных описаний таких переменных нет, то обращаться к ним по именам невозможно. Вместо этого к таким переменным обращаются через так называемые ссылочные имена (указатели), создаваемые при формировании динамических переменных. Ссылочный тип – это неограниченное множество значений, указывающих на элементы некоторого одного типа. Для ссылок не существует никаких операций кроме сравнения на равенство.

Пример однонаправленного связанного списка схематически можно представить в виде:



Указатель указывает на следующий элемент. Все элементы списка имеют одинаковый формат. Верхнее поле каждого элемента используется для указания следующего элемента. Если элемент не имеет следующего элемента, то содержащемуся в нем указателю присваивается значение **Nil.**

Описание типа «указатель» имеет вид:

^<Идентификатор типа>

Например,

```
type
  Link = ^object;
var
  P: Link ;
```

Переменная P является указателем на данные типа **object.** указатель – это адрес (в терминах программирования в кодах), а указываемая переменная – это содержимое, находящееся по адресу.

Пример 7.1. Описать динамическую структуру данных, которая является однонаправленным списком. Каждый элемент этой динамической структуры имеет два поля. Первое поле отводится под вещественное число, а второе – содержит адрес следующего элемента динамической структуры.

```
type
  Point = ^ Data ; Data = record
  R: real;
  Next: point
end;
var
  A : Point;
```

A – идентификатор, имеющий тип указателя. Вначале значение **A** не определено, так как описанная структура данных не создана и указатель не указывает адрес начального элемента динамической структуры. Чтобы создать такую структуру, в Паскале предусмотрена процедура **New(A).** Эта процедура создает первый элемент динамической структуры и помещает адрес этой структуры в указатель, т.е. в переменную **A**. Память для размещения элементов динамической структуры берется вне программы. Эту память принято называть «кучей», каждый элемент помещается в «кучу» в результате выполнения процедуры **New.** В связи с введением в Паскале типа данных «указатель» в стандарте языка предусмотрена предопределенная константа с именем **Nil. Nil** – это указатель, который ни на что не указывает, т.е. он имеет значение **Nil.**

В Паскале имеется процедура **Dispose,** форма записи которой имеет вид:

Dispose (идентификатор).

С помощью этой процедуры освобождается память, занятая структурами данных, на которые указывает ссылка. Освобожденная память возвращается в секцию **Heap** ("куча").

Например,

а)
```
var
  Q: ^t;
begin
```

```
…
…
New (Q);
…
…
Dispose (Q);
…
```

б)
```
type
  D1: ^ datal;
  D2 : ^ data2 ;
var
  El: D1;
  E2: D2;
```

Следующие ниже операторы присваивания запрещены в Паскале, так как E1 и E2 указатели разных типов данных:

El:=E2; или

E2:= El ;

После выполнения первого оператора присваивания окажется, что ссылка на объект типа **data1** будет ссылкой на объект типа **data2** и наоборот, что в Паскале запрещено.

Пример 7.2. Имеется файл целых положительных чисел, состоящий из нескольких последовательностей чисел, каждая из которых оканчивается целым отрицательным числом – 1. Вывести в выходной файл эти последовательности чисел, но внутри каждой последовательности числа должны идти в обратном порядке. 1 3 7 -1 2 0 6 7 -1 входной файл 7 3 1 -1 7 6 0 2 -1 выходной файл.

Длина этих последовательностей заранее неизвестна, поэтому переменные, в которые будут читаться эти числа должны иметь динамическую структуру.

```
Program Lists;
type
  Pointer=^DataSet;
  DataSet = record
    Data : integer
```

```
      Point: Pointer
   end;
var
   R1, R2: pointer;
   I: integer;
   Inp, Out: file of integer;
   F1, F2: file;
Begin
   Assign(F1, ‘example1’);
   Assign(F2, ‘example2’);
   Reset (F1) ;
   Rewrite (F2);
   while not Eof (F1) do
    begin
       Rl:=Nil;
       Read (F1, I);
       while I<>-1 do
        begin
           New (R2) ;
           R2^.data := I;
           R2^.point := R1;
           Rl:=R2;
           Read(F1, I)
        end;
       R2:=R1;
       while R2 <> Nil do
        begin
          Write(F2,R2^.data);
          R2:=R2^.point
        end;
       Write(F2,'-1')
    end
   Close(F1);
   Close(F2);
end.
```

Контрольные вопросы

1. Каково назначение переменных ссылочного типа?
2. Как распределяется память под переменные ссылочного типа?
3. Каково назначение процедур NEW и DISPOSE, MARK и RELEASE?
4. В чем состоит отличие механизмов работы этих процедур?
5. Дайте определение динамической переменной?
6. Чем отличается динамическая переменная от статической?
7. Что понимается в языке Паскаль под кучей?
8. Какие операции выполняются над переменными ссылочного типа?
9. Как организуется однонаправленный список?
10. Каким образом можно исключить элемент из списка?
11. Как организуется вставка элемента в список?
12. Каким образом можно добавить элемент в конец списка?

Задание к работе

Выполнить индивидуальное задание.

Методические указания

1. Необходимо ознакомиться с примером программы SPISOK.
2. При решении задачи использовать ссылочный тип данных.
3. Разработать алгоритм решения задачи.
4. Написать программу.
5. Отладить программу.
6. Разработать тестовые наборы данных на проверку полноты функционирования программы.
7. Оформить отчет и написать выводы по эффективности использования ссылочных типов данных.

Содержание отчета

1. Титульный лист.
2. Словесная постановка задачи.
3. Графический или текстуальный алгоритм решения задачи.
4. Листинг программы.
5. Контрольный тест и результаты тестирования программы.
6. Ответы на контрольные вопросы.

Варианты индивидуальных заданий

1. Организовать однонаправленный список всех простых чисел, меньших n. Удалить из списка элементы, значения которых лежат в диапазоне от $m1$ до $m2$. Результаты обработки списка вывести на экран.

2. Дано натуральное число n (n>2). Найти все меньшие n простые числа, используя решето Эратосфена. Решетом Эратосфена называют следующий способ. Выпишем подряд все целые числа от 2 до n. Первое простое число 2. Подчеркнем его, а все большие числа, кратные 2, зачеркнем. Первое из оставшихся чисел 3. Подчеркнем его как простое, а все большие числа, кратные, 3, зачеркнем. Первое число из оставшихся теперь 5, так как 4 уже зачеркнуто. Подчеркнем его как простое, а все большие числа, кратные 5 зачеркнем и т.д.: 2, 3, 4, 5, 6, 7, 8, 9, 10, ... Исходную последовательность чисел организовать в виде однонаправленного списка. Удаление производить внутри этого списка, не используя дополнительные списки.

3. Дана действительная матрица размера nxm; организовать однонаправленный список строк матрицы, упорядоченных:

а) по не убыванию значений первых элементов строк.

б) по не возрастанию значений наибольших элементов строк.

4. Даны действительные числа a_1,....., a_n, p, натуральное число k, так что ($a_1 < a_2 < ... < a_n$, $k < n$). Удалить из последовательности a_1, ..., a_n элемент с номером k (то есть a_k) и вставить элемент, равный p, так чтобы не нарушилась упорядоченность. Последовательность a_1,....., a_n представить в виде однонаправленного списка.

5. Дана действительная матрица размера nxm; организовать однонаправленный список строк матрицы, упорядоченных:

а) по не возрастанию сумм элементов строк.

б) по не убыванию значений наименьших элементов строк.

6. Таблица выигрышей денежной лотереи представлена массивом выигрышных номеров a_1, ..., a_n и массивом выигрышей в рублях p_1, ..., p_n (p_i – это выигрыши, выпавший на номер a_i). Разместите эту таблицу в динамической области памяти в виде однонаправленного списка. Организовать удаление элементов списка по мере получения выигрышей. Результаты обработки выводить на экран.

7. Дано натуральное число n. Среди чисел 1, ..., n найти все такие, запись которых совпадает с последними цифрами записи их квадрата, напри-

мер $6^2 = 36$, $25^2 = 625$ и т.д. Представить их в виде однонаправленного списка, элементами которого являются само число и его квадрат.

8. Пусть дан массив a_1, ..., a_n. Требуется переставить a_1, ..., a_n, так чтобы в начале в массиве шла группа элементов больших того элемента, который в исходном массиве располагался на первом месте, затем – сам этот элемент, потом – группа элементов, меньших или равных ему. Использовать однонаправленный список.

9. Назовем натуральное число палиндромом, если его запись читается одинаково с начала и с конца, например 4884, 393, 1. Найти все меньшие 100 натуральные числа, которые при возведении в квадрат дают палиндром. При решении задачи используйте двунаправленный список.

10. Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенных в n-ую степень равна самому числу, например, $153 = 1^3 + 5^3 + 3^3$. Получить все числа Армстронга, состоящие из двух, трех и четырех цифр, организовать соответственно 3 однонаправленных списка.

3. КРАТКАЯ СПРАВКА

Процедуры и функции ввода-вывода

BlockRead Считывает из нетипизированного файла одну или более записей.
BlockWrite Записывает в нетипизированный файл одну или более записей.
ChDir Выполняет смену текущего каталога.
Erase Стирает внешний файл.
Eof Возвращает для файла состояние end-of-file (конец файла).
FilePos Возвращает текущую позицию в файле. Для текстовых файлов не используется.
FileSize Возвращает текущий размер файла. Для текстовых файлов не используется.
Flush Сбрасывает буфер текстового файла вывода.
Getdir Возвращает текущий каталог на заданном диске.
IOResult Возвращает целое значение, являющееся состоянием последней выполненной операции ввода-вывода.
MkDir Создает подкаталог.
Rename Переименовывает внешний файл.
Reset Открывает существующий файл.

Rewrite Создает и открывает новый файл.
RmDir Удаляет пустой подкаталог.
Seek Перемещает текущую позицию в файле на заданный элемент. Для текстовых файлов не используется.
SeekEof Возвращает для текстового файла состояние "конец файла".
SeekEoln Возвращает для текстового файла состояние "конец строки".
SetTextBuf Назначает для текстового файла буфер ввода-вывода.
Truncate Усекает размер файла до текущей позиции. Для текстовых файлов не используется.

СПИСОК ЛИТЕРАТУРЫ

1. Аврамов В. П., Епифанов В. В., Медведев Н. Г. Методические указания по тяговому расчету транспортной гусеничной машины с дизельным двигателем и механической ступенчатой трансмиссией. – Харьков: ХПИ, 1991. – 36с.

2. Александров Е. Е., Волонцевич Д. О., Лебедев А. Г. и др. Динамика транспортно–тяговых колесных и гусеничных машин. – Харьков, 2001. – 640 с.

3. Епанешников А. М., Епанешников В. А Программирование в среде Turbo Pascal 7.0. – 3-е изд. стереотип. – М.: ДИАЛОГ МИФИ, 1998. – 282с.

4. Зуев Е. А. Программирование на языке Turbo Pascal 6.0,7.0. – М.: Веста: Радио и связь, 1993. – 304с.

5. Зуев Е. А. Turbo Pascal. Практическое программирование. – М.: Стрикс, 1997. – 334с.

6. Турбо Паскаль 7.0 – К.: Издательская группа BHV, 1996. – 448с.

7. Зубов В. С. Программирование на языке TURBO PASCAL (версии 6.0 и 7.0). Изд. 2-е, перераб. и доп. – М.: Информационно-издательский дом «Филинъ», 1997. – 320с.

8. Довгаль С.И., Литвинов Б.Ю., Сбитнев А.И. Персональные ЭВМ: ТурбоПаскаль V 6.0. Объектное проектирование. Локальные сети. – Киев: «Информсистема сервис», 1993. – 440 с.

9. Лабораторный практикум по программированию на языке Паскаль: Учебное пособие. / Под общ. ред. Л. В. Найхановой и Н. Ц. Бильгаевой. – изд. 3-е перераб. и доп., – Улан-Удэ, 2004. – 176 с.

Навчальне видання

Методичні вказівки к лабораторним заняттям з курсу „Основи програмування” для студентів спеціальності 7.080402 „Інформаційні технології проектування”

Російською мовою

Укладачі: **ВОРОНЦОВ** Сергій Миколайович
КОСТЯНИК Ірина Віталіївна
СЄРИКОВ Володимир Іванович

Відповідальний за випуск М. А. Ткачук
Роботу рекомендував до видання В. К. Бєлов

В авторській редакції

План 2007 р., поз. 49/

Підписано до друку __.__.__. Формат 60×84 1/16. Папір офсетний.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 2,5.
Обл. – вид. арк. 3,0. Наклад 100 прим. Зам № . Ціна договірна.

Видавничий центр НТУ „ХПІ”, 61002 Харків, вул. Фрунзе, 21
Свідоцтво про державну реєстрацію ДК №116 від 10.07.2000 р.

Друкарня НТУ „ХПІ”, 61002 Харків, вул. Фрунзе, 21